nature portfolio

Peer Review File

OCTOPUS: Operation Control System for Task Optimization and Job Parallelization via a User-Optimal Scheduler

Corresponding Author: Dr Sang Soo Han

This file contains all reviewer reports in order by version, followed by all author rebuttals in order by version.

Version 0:

Reviewer comments:

Reviewer #1

(Remarks to the Author)
This is the referee report for
OCTOPUS: Operation Control System for Task Optimization and Job Parallelization via a User-Optimal Scheduler by Hyuk Jun Yoo, Kwan-Young Lee, Donghun Kim, and Sang Soo Han

This paper introduced the system of OCTOPUS: operation control system for task optimization and job parallelization via a user-optimal scheduler. I understood the importance of such job scheduling applications for the automation laboratory, and this is great work from the point of view of technology. In particular, the authors focus on the job waiting time, and it can be reduced via the system. On the other hand, this paper lacks the scientific finding or real materials developments. Thus, I believe that this manuscript is not suitable for Nature Communications. By controlling the real experiments using the proposed system, if the interesting materials are developed, it would fall under the scope of Nature Communications. I recommend that the current manuscript be submitted to a journal specializing in “methods”.

Reviewer #2

(Remarks to the Author)
Dear editor,

I have reviewed the manuscript titled "CTOPUS: Operation Control System for Task Optimization and Job Parallelization via a User-Optimal Scheduler ". Overall, the manuscript provides a comprehensive overview of the OCTOPUS platform and its innovative features for modular automated experimentation. The authors have done an excellent job of detailing the various techniques employed within OCTOPUS, such as job parallelization, task optimization, and the closed-packing schedule algorithm, to optimize experimental processes and resource utilization.

However, I believe that further exploration of the integration of extended reality (XR) technologies, such as virtual reality (VR) and augmented reality (AR), could significantly enhance the manuscript. The authors briefly mention the potential benefits of incorporating XR technologies into the OCTOPUS platform but do not delve into the feasibility, practical implications, and challenges associated with XR integration in modular automated platforms (MAP) systems.

Specifically, I recommend that the authors expand upon the following points in their manuscript:

1. Feasibility of XR Integration: The authors should discuss the technical feasibility of integrating VR and AR technologies into the OCTOPUS platform. This discussion should include considerations such as hardware requirements, software development, and compatibility with existing experimental setups.

2. Practical Implications: Explore the practical implications of incorporating XR technologies in experimental setups. Discuss how VR and AR interfaces could improve user experience, facilitate remote collaboration, and enhance experimental planning and execution within OCTOPUS.

3. Potential Benefits: Provide a detailed analysis of the potential benefits of XR integration in MAP systems. Consider how

immersive VR environments could simulate experimental processes, while AR overlays could provide real-time data visualization and guidance during experiments.

4. Challenges and Limitations: Address the challenges and limitations associated with XR integration, such as cost, usability, and integration with existing laboratory equipment. Discuss potential strategies for overcoming these challenges and mitigating any associated risks.

5. By addressing these points, the authors can enrich their manuscript with valuable insights into the future directions of modular automated experimentation and the role of XR technologies in advancing research in materials science and related fields.

Overall, I recommend that the authors revise their manuscript to include a dedicated section on XR integration, providing a thorough exploration of its feasibility, practical implications, benefits, challenges, and potential solutions.

Thank you for considering my recommendations.

Reviewer #3

(Remarks to the Author)
1) What are the noteworthy results?

The authors have written code to break down chemistry experiments into subtasks, schedule those subtasks to be executed in a way that optimizes for resource utilization, and triggers the subtasks to be executed. The software is capable of handling multiple users and multiple lab resources.

2) Will the work be of significance to the field and related fields? How does it compare to the established literature? If the work is not original, please provide relevant references.

What the authors have done is a challenging engineering task and a useful step forward for their lab; however, the work is not novel in the context of scientific publications. Please see my code comments regarding its reusability.

UC Berkeley has an autonomous lab that has been used for automated materials science experiments (https://www.nature.com/articles/s41586-023-06734-w).

Berkeley has also recently released a pre-print on workflow management software for autonomous labs (https://arxiv.org/abs/2405.13930).

Caltech also has the capacity to orchestrate automated experiments across a variety of lab equipment, and they have published papers on the code used to create, schedule, and trigger subtasks (https://pubs.rsc.org/en/content/articlehtml/2023/dd/d3dd00166k).

3) Does the work support the conclusions and claims, or is additional evidence needed?

Yes, I believe that they have accomplished what they claim. They have provided the code and video evidence of the code working in their lab in addition to a detailed description of how it works.

4) Are there any flaws in the data analysis, interpretation and conclusions? Do these prohibit publication or require revision?

No, I don't see flaws in the conclusions. They have created what they say they have created.

5) Is the methodology sound? Does the work meet the expected standards in your field?

Please see my comments below regarding the code review.

6) Is there enough detail provided in the methods for the work to be reproduced?

Yes. The work would be difficult to reproduce because it requires a robotic lab setup. Also, more documentation, code comments, and type annotations would be useful, but I don't see any outright missing information or anything else that the authors should be required to provide before publishing.

Version 1:

Reviewer comments:

Reviewer #4

(Remarks to the Author)

This work is a tour de force in establishing a code base coupled with detailed demonstrations of its deployment for automated materials science experiments. The authors describe various complementary works from the community, although I don't believe any published work covers the breadth of the present submission. I recommend this work for publication with very minor revisions, which are noted below after my assessment of how the reviewers addressed the comments from the previous review:

. Reviewer 1 has noted the lack of "scientific findings", although I see this paper as a clear outline of how most scientific findings will be generated in labs that fully leverage the power of automation and AI. I think this work is quite visionary and I can speak firsthand on the technical difficulty of developing such a system. I think the combination of technical advance and vision make the work suitable for publication in Nat Comm.

. Reviewer 2 noted the potential to improve the manuscript via discussion how "extended reality" may be leveraged in OCTOPUS. The authors nicely implemented this suggestion.

. The comments of Reviewer 3 have been addressed via cleanup of the git repository, addition of the copilot, implementation of pydantic models, and implementation of Auth0.

. I want to especially note that the addition of the co-pilot elevates the work compared to the initial submission. There has been lots of buzz around the use of LLMs for experiment design, and this work shows how the OCTOPUS infrastructure can capitalize on this opportunity.

2 minor suggestions:

. In the abstract when the copilot is introduced with the phrase "ensure the reusability of OCTOPUS", I think "ensure" is an overstatement and can be replaced with, for example, "promote".

. For the "Network protocol-aided process modularization" in Fig. 4. To my knowledge, the first demonstration of this type of functionality was the following paper, which the authors may consider discussing and citing:
Vogler, M.; Busk, J.; Hajiyani, H.; Jørgensen, P. B.; Safaei, N.; Ramírez, F. F.; Carlsson, J.; Pizzi, G.; Clark, S.; Bhowmik, A.; Stein, H. S. Brokering between Tenants for an International Materials Acceleration Platform.

Responses to Reviewer Comments

Reviewer 1

[**General review**] “This is the referee report for OCTOPUS: Operation Control System for Task Optimization and Job Parallelization via a User-Optimal Scheduler by Hyuk Jun Yoo, Kwan-Young Lee, Donghun Kim, and Sang Soo Han.

This paper introduced the system of OCTOPUS: operation control system for task optimization and job parallelization via a user-optimal scheduler. I understood the importance of such job scheduling applications for the automation laboratory, and this is great work from the point of view of technology. In particular, the authors focus on the job waiting time, and it can be reduced via the system.

On the other hand, this paper lacks the scientific finding or real materials developments. Thus, I believe that this manuscript is not suitable for Nature Communications. By controlling the real experiments using the proposed system, if the interesting materials are developed, it would fall under the scope of Nature Communications. I recommend that the current manuscript be submitted to a journal specializing in “methods”.”

[Our response] We appreciate the reviewer's insightful evaluations of our work. OCTOPUS is an innovative software that significantly enhances the functionality of the Material Acceleration Platform (MAP). Previously limited to handling single experiments, MAP can now schedule and manage multiple experiments concurrently with the system integration via OCTOPUS, thereby substantially increasing the number of experiments conducted. Although our current study does not present new material developments, we firmly believe that the combined capabilities of MAP and OCTOPUS will greatly accelerate the future development of various materials.

Reviewer 2

[**General review**] “I have reviewed the manuscript titled "OCTOPUS: Operation Control System for Task Optimization and Job Parallelization via a User-Optimal Scheduler ". Overall, the manuscript provides a comprehensive overview of the OCTOPUS platform and its innovative features for modular automated experimentation. The authors have done an excellent job of detailing the various techniques employed within OCTOPUS, such as job parallelization, task optimization, and the closed-packing schedule algorithm, to optimize experimental processes and resource utilization.

[Our response] We thank the reviewer for the positive evaluation of our work. We added discussions related to the below **Comment 1** in the manuscript.

[Comment 1] “However, I believe that further exploration of the integration of extended reality (XR) technologies, such as virtual reality (VR) and augmented reality (AR), could significantly enhance the manuscript. The authors briefly mention the potential benefits of incorporating XR technologies into the OCTOPUS platform but do not delve into the feasibility, practical implications, and challenges associated with XR integration in modular automated platforms (MAP) systems. Specifically, I recommend that the authors expand upon the following points in their manuscript:

1. Feasibility of XR Integration: The authors should discuss the technical feasibility of integrating VR and AR technologies into the OCTOPUS platform. This discussion should include considerations such as hardware requirements, software development, and compatibility with existing experimental setups.

2. Practical Implications: Explore the practical implications of incorporating XR technologies in experimental setups. Discuss how VR and AR interfaces could improve user experience, facilitate remote collaboration, and enhance experimental planning and execution within OCTOPUS.

3. Potential Benefits: Provide a detailed analysis of the potential benefits of XR integration in MAP systems. Consider how immersive VR environments could simulate experimental processes, while AR overlays could provide real-time data visualization and guidance during experiments.

4. Challenges and Limitations: Address the challenges and limitations associated with XR integration, such as cost, usability, and integration with existing laboratory equipment. Discuss potential strategies for overcoming these challenges and mitigating any associated risks.

By addressing these points, the authors can enrich their manuscript with valuable insights into the future directions of modular automated experimentation and the role of XR technologies in advancing research in materials science and related fields. Overall, I recommend that the authors revise their manuscript to include a dedicated section on XR integration, providing a thorough exploration of its feasibility, practical implications, benefits, challenges, and potential solutions.”

[Our response] We appreciate the constructive feedback on XR technologies. The integration of XR (Extended Reality), which includes both VR (Virtual Reality) and AR (Augmented Reality), holds significant potential to greatly enhance the accessibility and usability of OCTOPUS.

VR[1,3] is defined as a technology that immerses users in a completely virtual environment. Recent advance[2] show the potential implementation of VR in MAP. The benefits of VR for MAP include replicating client hand motions via remote control. Therefore, VR can be employed for recovery motion planning in the event of robotic action failures, helping to prevent safety accidents and democratizing client interaction by offering surveillance-free MAP.

Additionally, AR[3] is defined as a technology that overlays virtual elements onto the real-world environment. In MAP, the AR technology with its advanced visualization capabilities can aid in decision-making by visualizing job status, issuing warning alerts, displaying AI decision processes, and providing knowledge graphs of accumulated data.

However, XR technologies are highly device-dependent, resulting in substantial differences in price, performance, and interfacing methods among various manufacturers[3]. Furthermore, there is a notable lack of real-time data processing infrastructure for XR devices. These technical bottlenecks hinder the implementation of XR technologies within MAP today. Therefore, there is a pressing need for standardized interfacing protocols to integrate heterogeneous XR devices, as well as the development of real-time data processing capabilities utilizing 4G/5G networks and blockchain-based encrypted communication infrastructure.

[Revision to the manuscript] Following the reviewer's suggestion, we modified discussion parts to reflect Copilot of OCTOPUS.

(Discussion- p16, 17) "Moreover, the integration of XR (extended reality), including VR[48] (virtual reality) and AR[49] (augmented reality), holds significant potential to greatly enhance the accessibility and usability of OCTOPUS. [15,50–52] VR is defined as a technology that immerses users in a completely virtual environment, and recent advancements show the potential implementation of VR in MAP. The benefits of VR for MAP include replicating client hand motions via remote control. Thus, VR can be utilized for recovery motion planning in the event of robotic device failures. The implementation of VR helps prevent safety accidents and democratizes client interactions through surveillance-free MAP.

Additionally, AR is defined as a technology that overlays virtual elements onto the real-world environments. In MAP, the AP technology with its specialized visualization capabilities, can assist in decision-making by displaying job status, safety alerts, AI decision processes, and knowledge graphs of accumulated data.

However, XR technologies are highly device-dependent, leading to considerable variation in price, performance, and interfacing methods across different manufacturers. Additionally, there is a notable lack of real-time data processing infrastructure for XR devices. These technical bottlenecks hinder the implementation of XR technologies within MAP today. Therefore, standardized interfacing protocols for integrating heterogeneous XR devices are needed, along with the development of real-time data processing capabilities using 4G/5G networks and blockchain-based encrypted communication infrastructure."

[References]

[1] Skibba, Ramin. "Virtual reality comes of age." *Nature* 553.7689 (2018): 402-403.

[2] Li, Jiagen, et al. "Toward “on-demand” materials synthesis and scientific discovery through intelligent robots." *Advanced Science* 7.7 (2020): 1901957.
[3] Matthews, David. "Virtual-reality applications give science a new dimension." *Nature* 557.7703 (2018): 127-128.

Reviewer 3

[Comment 1] “In this case, completely reproducing what they have done in the paper would involve having access to their robotic experiment setup. So instead, I downloaded the code, attempted to install it, and read through it to understand how it works.

Regarding installation, it is easy to git clone their repo, but the requirements.txt that they mention in the README is not present, which makes it difficult to set up an appropriate python environment to run their code.”

[Our response] We thank you for your precise comments. We acknowledge the error in our file upload and appreciate the opportunity to correct it. We have uploaded the latest version of the **requirements.txt** files to our GitHub repository for virtual environment installation (https://github.com/KIST-CSRC/Octopus).

[Comment 2] “1) Hard-coded use-case-specific information.

For example, the files in the TaskAction folder contain a great deal of information that is specific to the use case presented. Specifically, for example, methods like "stir," "mix," and "react" exist, and the innerworkings of those functions depend on specific dictionary keys being present.

Since the authors are publishing a paper on the code specifically, I assume they intend for it to be re-usable by others. If someone else wanted to use the OCTOPUS system for a new set of lab resources, it would not be clear which files and functions need to be edited, and which should be left alone.

Ideally, the functionality would be separated from the use case, and a clear line would be drawn between the core software and the use-case-specific information that should be edited by a future user.”

[Our response] Thank you to the reviewer for this comment. Following the reviewer's suggestion, we acknowledge and agree with the assessment regarding the low reusability of OCTOPUS when MAP scales up. When a client registers a new module to OCTOPUS, the unique robotic-lab settings necessitate hands-on code generation and customization for integration into the system. According to Table R1, the traditional OCTOPUS system exhibits low reusability, requiring code generation and customization for several components, including the task generator, action translator, action executor, resource manager, and module node. Additionally, manual code generation and customization can introduce operational errors into OCTOPUS due to human errors, such as using invalid data types or incorrectly defining functions that do not follow the established format. This complex module registration process can impede the flexibility and scalability of OCTOPUS. To address these challenges, we developed the **Copilot of OCTOPUS**, enabling convenient module registration through automated code generation.

O = Reusable
▲= Partial modification required
X = Complete modification required

Component	Reusability without Copilot of OCTOPUS	Reusability with Copilot of OCTOPUS
Job scheduler	O	O
Task generator	X	O
Task scheduler	O	O
Action translator	X	▲ (considered robotic-lab setups)
Action executor	▲	O
Resource manager	X	O
Module node	X	▲ (considered robotic-lab setups)

Table R1. Reusability comparison without and with Copilot of OCTOPUS. The circle represents reusable code for core software. The triangle represents semi-reusable code that requires partial modifications based on the robotic-lab setups of the module. The X mark represents non-reusable code that needs to be completely modified according to the robotic-lab setups of the module.

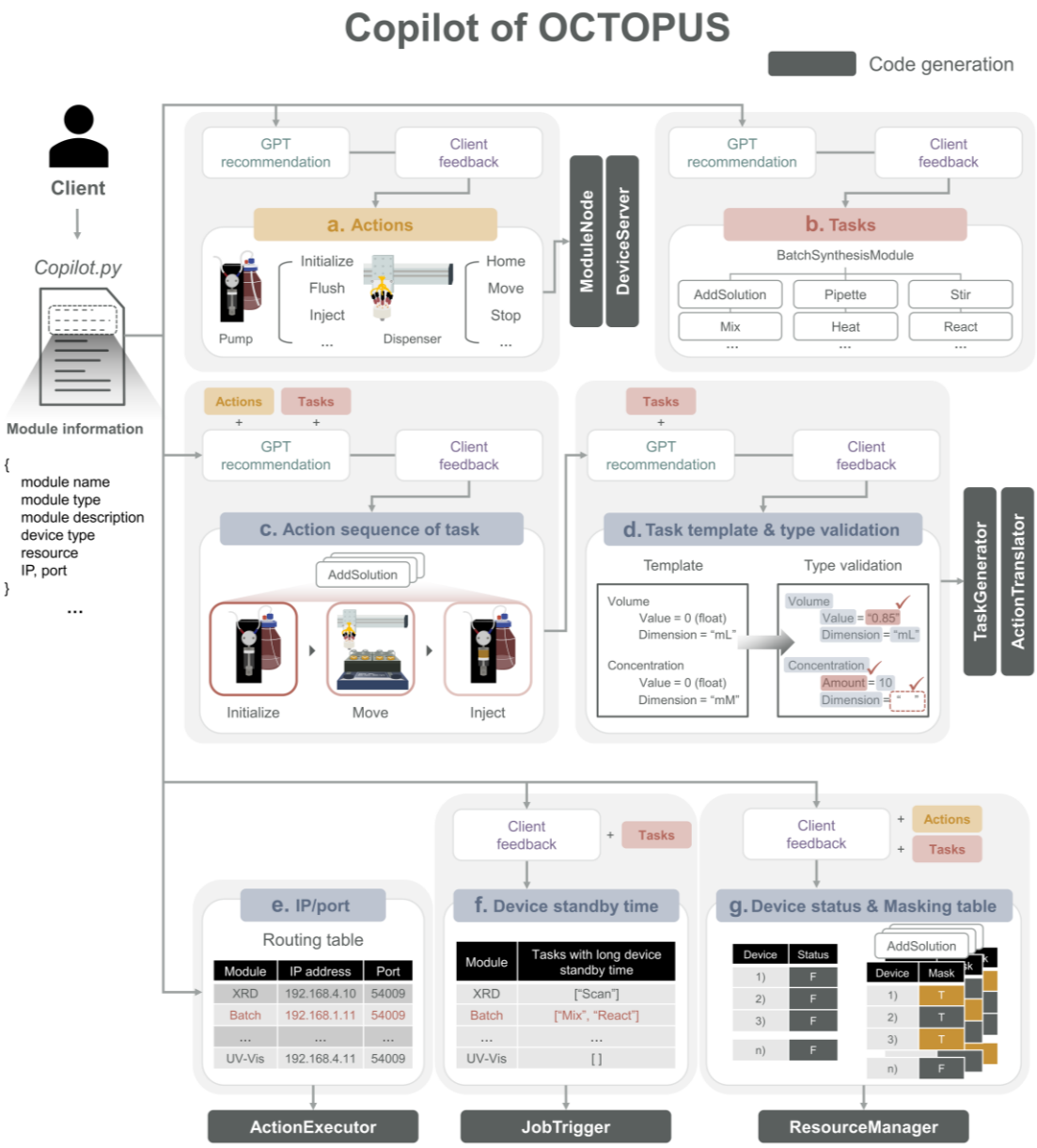


Figure R1. Copilot of OCTOPUS. Copilot of OCTOPUS consists of seven steps for code generation and customization in OCTOPUS. Gray boxes represent the generated codes. **(a)** Action generation: Device actions are facilitated through GPT recommendations and a client feedback system. Code generation occurs within the module node and device server. **(b)** Task generation: Module tasks are generated through GPT recommendations and the client feedback system. **(c)** Action sequence generation: Action sequences for tasks are generated through GPT recommendations and the client feedback system. Before the GPT modeling process begins, the generated tasks and actions are added to the GPT prompt. **(d)** Task template and type validation: Task templates and type validations are generated through GPT recommendations and the client feedback system. Before the GPT modeling process begins, the generated tasks are added to the GPT prompt. Code generation takes place in the task generator and action translator. **(e)** IP address and port number registration: This step establishes connections with the module node. Code generation occurs in the action executor. **(f)** Task registration with long device standby time: Tasks with long device standby times are registered through the client feedback system using the generated tasks. Code generation takes place in the job trigger. **(g)** Device registration: New devices are registered in the device status table and masking table for task execution through the client feedback system, using the generated tasks and actions. Code generation takes place in the resource manager.

Copilot of OCTOPUS features GPT (generative pre-trained transformer)-based recommendations and client feedback to streamline module generation and validation. In the GPT prompt design, few-shot learning with several example cases was performed to enhance its prediction accuracy. Copilot of OCTOPUS is supposed to auto-generates required codes and files to run OCTOPUS, and the process is as follows. The first step is that a client needs to input simple module information in the **'copilot.py'** Python file. The execution of '**copilot.py**' generates a list of actions and tasks based on the input module information through GPT recommendations, then the client reviews and modifies the list (Figs. R1a and R1b), ensuring the adaptation to diverse applications and user preferences. Next, the action sequences for a specific task as well as task templates are generated via GPT recommendations and client feedback (Figs. R1c and R1d). In this stage, the data type validations are performed to check for invalid data in the task template, using "Pydantic" to predefine the JSON data type. Lastly, it updates the routing table for managing IP addresses and ports for different modules, identifies the tasks with long device standby times, and completes the device status table and masking tables with client feedbacks. As a result of these sequential processes, several codes are auto-generated, as described in Table R2. We provide **Supplementary Guideline** for clients to use Copilot of OCTOPUS without difficulty. **New users can now easily generate diverse codes for new modules via Copilot of OCTOPUS, requiring only minimal modifications to 'copilot.py'.**

We agree with the reviewer's comments that OCTOPUS needs to separately present core software and use-case-specific software more clearly, and thus we do so here. Copilot of OCTOPUS is divided into core software and use-case-specific software. Purple boxes in Table R2 represent the scripts of core software that do not require code modifications, regardless of new module registration. We categorize use-case-specific software based on module operation and module setting information. Module operations that require specific logic, such as translating tasks into actions in the action translator or allocating resources in the resource manager, must be newly defined during module registration and are saved separately as new script files, as indicated by the green boxes in Table R2. Additionally, new module information is addressed in the JSON file during code generation, as shown by the red boxes in Table R2. More detailed results of automated code generation are described in Table R2.

1st hierarchy	2nd hierarchy	3rd hierarchy		Description
Action	Module	{module name}.py		Action translator script of module
	routing_table.json			JSON file included IP addresses and port number of each module
	ActionExecutor_Class.py			Script of action executor
	ActionTranslator_Class.py			Script of action translator
Analysis	Module	{module name}.py		Preprocess script for raw spectrum or performance data of each module
	Analysis.py			Script of analysis method inherited by Module/{module name}.py
Algorithm	Manual	Manual.py		Script of manual experimentation method
	{algorithm name}	{algorithm name}.py		Script of AI for experiment planning
AutoModule Generation	GPT_answer_ generation	{module name} _actions.txt		GPT recommendations for actions and tasks generation saved as text files
	GPT_answer_registration	{module name} _actionsequence.txt		GPT recommendations for action sequences saved as text files
		{module name} _task_pydantic.txt		GPT recommendations for task template generation saved as text files
		{module name} _task_template.txt		GPT recommendations for type validation of task (=Pydantic) saved as text files
	{module name or device server name}	BaseUtils	json_func.py	Functions for socket communications
			TCP_Node.py	Functions for JSON file read/write
		Log	Logging_ Class.py	Logger for recording all actions
		{device_name}		Functions for device controller based on manufacturer's API (Application Programming Interface)
		module_node.py or device_server.py		Interface of module node or device server
DB	DB_Class.py			Script of MongoDB manager
Job	device_standby_time.json			JSON file included tasks with long device standby time for each module
	JobTrigger.py			Script of job trigger
	JobScheduler.py			Script of job scheduler
JobScript Template	{module name}.json			Template of job script for each module
Log	Logging_Class.py			Script of logger
Resource	Module	{module name}.py		Resource allocator for each module
	device_location.json			Device resources (location information) for each module
	device_status.json			Device status table included all devices for each module
	device_masking_table.json			Device masking table for all task of each module
	ResourceAllocator_Class.py			Script of resource allocator inherited by Module/{module name}.py
	ResourceManager_Class.py			Script of resource manager
Task	ActionSequence	{module name}.json		JSON file included action sequence of tasks for each module
	Pydantic	{module name}.py		Script of type validation for each module via Pydantic library
	Template	{module name}.json		JSON file included task templates for each module
	Template_module.json			JSON file included all module templates
	TaskGenerator_Class.py			Script of task generator
	TaskScheduler_Class.py			Script of task scheduler
USER	{client ID}			
UserManager	Auth0_config.py			Script of Auth0 configuration for login process
	UserManager_Class.py			Script of Auth0 functions for login process
client.py				Script of client
copilot.py				Script of Copilot of OCTOPUS
master_node.py				Script of master node

* {}: the real name of module, ex) {module name}="SolidStateModule"

Table R2. Code hierarchy of OCTOPUS with descriptions. Purple boxes represent the core software of OCTOPUS. Green boxes indicate the automated code generation for module operation via Copilot of OCTOPUS, which includes functions for module operation. Red boxes represent the JSON file addressing new module information via Copilot of OCTOPUS.

[Revision to the manuscript] To describe Copilot of OCTOPUS, we added a new figure (**Fig. 9**) and also modified abstract, introduction, result and discussion parts accordingly in the manuscript.

A **Supplementary Guideline** is appended to this article, providing a detailed and step-by-step protocol to assist potential clients in using Copilot of OCTOPUS without difficulty.

We also provide new items of Figures S30-S34 and Tables S3 and S4 in **Supplementary Information**.

(Abstract) "... Copilot of OCTOPUS is developed to ensure the reusability of OCTOPUS for potential users with their own sets of lab resources, which substantially simplifies the process of code generation and customization through GPT recommendations and client feedback."

(Results and Discussion – p15-16)

"Copilot of OCTOPUS. If a client wants to use the OCTOPUS system for a new set of lab resources, it will require a lot of hands-on code generation and customization for the integration within OCTOPUS. A partial or complete modification is required for generating components of task generator, action translator, action executor, resource manager, and module node, as illustrated in Supplementary Figure S30. To improve the reusability of OCTOPUS, we developed "Copilot of OCTOPUS" to offer convenient module registration process via automated code generations (Fig. 9). Copilot of OCTOPUS features GPT (generative pre-trained transformer)-based recommendations and client feedback to streamline module generation and validation. In the GPT prompt design, a few-shot learning with several examples was performed to enhance its prediction accuracy, and more details on prompt engineering are described in Supplementary Figures S31-S34.

Copilot of OCTOPUS auto-generates required codes and files to run OCTOPUS, and the process is as follows. The first step is that a client needs to input simple module information in the 'copilot.py' Python file. The execution of 'copilot.py' generates a list of actions and tasks based on the input module information through GPT recommendations, then the client reviews and modifies the results (Figs. 9a, 9b and Supplementary Figures S31 and S32), ensuring the adaptation to diverse applications and user preferences. Next, the action sequences for a specific task as well as task templates are generated via GPT recommendations and client feedback (Figs. 9c, 9d and Supplementary Figure S33). In this stage, the data type validations are generated to check for invalid data in the task template, using Pydantic[46] to predefine the JSON data type (Fig. 9d and Supplementary Figure S34). Lastly, it updates the routing table for managing IP addresses and ports for different modules, identifies the tasks with long device standby times, and completes the device status table and masking tables with client feedbacks. As a result of these sequential processes, all required codes and files are auto-generated, as described in Supplementary Table S4. New users can now easily generate diverse codes for a new set of lab resources via Copilot of OCTOPUS, requiring only minimal modifications to 'copilot.py'. A Supplementary Guideline is appended to this article, providing a detailed, step-by-step protocol to assist potential clients in using the Copilot feature of OCTOPUS without difficulty.

Copilot of OCTOPUS automates the code generation and customization for new module registration, requiring only simple input regarding module information from clients. Copilot of OCTOPUS streamlines operations across different lab setups by reducing the need for manual adjustments, minimizing human error, and enabling seamless task execution (Supplementary Figures S30 and S34). This automated process not only enhances operational efficiency for module registration but also ensures the accuracy and reliability of module integration within OCTOPUS. The significantly enhanced reusability of OCTOPUS supports scalable MAP evolution, and meets growing industrial demands.

Although Copilot of OCTOPUS simplifies the process of code generation and customization through GPT recommendations and client feedback, the accuracy of these recommendations is significantly influenced by the predefined prompts. Recent advancements[47] in prompt engineering have demonstrated that well-customized initial prompts can enhance the quality of GPT responses. To achieve accurate GPT recommendations, further research into advanced prompt engineering for Copilot of OCTOPUS is necessary."

"

(Results and Discussion – p17) "…Copilot of OCTOPUS is provided to ensure the reusability of OCTOPUS for potential users, which significantly simplifies the process of code generation and customization with GPT recommendations and client feedbacks."

(Supplementary Guideline) *– A file providing a detailed and step-by-step protocol to assist potential clients in using Copilot of OCTOPUS without difficulty.*

(Supplementary Information – Figures S30-S34, Table S3 and S4)

Supplementary Figure S30. Reusability comparison with and without Copilot of OCTOPUS

Supplementary Table S3. The description of inputs and outputs of GPT

Supplementary Figure S31. Example of action generation for each device in module

Supplementary Figure S32. Example of task generation for module

Supplementary Figure S33. Example of the action sequence generation for task execution

Supplementary Figure S34. Example of task template and type validation generation

Supplementary Table S4. The result of automated code generation/customization via the Copilot of OCTOPUS

[Comment 3] “2) Lack of typing and data validation

In many places in the code (for example react_time = task_dict["Time"]["Value"]), it is clear that the code expects certain dictionary keys to be present, but the schema for these dictionaries is not explicitly stated, which would make it difficult for a future user to know what the format of the dictionaries should be. Ideally, the authors would use pydantic or a similar package to explicitly state the format of each dictionary. Then, it would be possible to validate dictionaries and use type annotations to convey which data types need to be passed to which functions.

Additionally, the database is a schemaless document database with no data validation at the application level or database level. Therefore, if multiple users sent data to this database, it is very likely that a compilation of differently-formatted data would accumulate, and it would be difficult for a researcher to query the database and make sense of the results.”

[Our response] We thank the reviewer for this comment. The reviewer comment made us to recognize the limitations of the current OCTOPUS code system, particularly the lack of typing and data validation. To address this, we implemented the Pydantic[1] package to explicitly define the format of each task template, ensuring clear and precise task template statements.

Copilot of OCTOPUS can automatically generate Pydantic classes for task templates. In Figure R2, role assignment, prompts, and examples improve the accuracy of code generation for Pydantic classes through few-shot learning. The role assignment section defines the system's role in controlling MAP for chemical experiments, focusing on Pydantic generation. The prompt section outlines the task prompt, instructing the definition of Pydantic classes for various batch synthesis modules, such as "MoveContainer," "AddSolution," "Stir," "Heat," "Mix," and "React." The examples for the few-shot learning section provide sample code snippets, demonstrating the structure and implementation of Pydantic classes for different module parameters, aiding the learning process for generating accurate models.

Role assignment

```
"role": "system",
"content": """You are administrator of autonomous laboratory, which control AI and robotics for chemical experiments,
and your role is the Pydantic generation for chemical task.""",
```

Prompt

```
task_pydantic_prompt="""Also, define a pydantic class followed by json file, and don't skip some task, just define all of it.

if we set task list as BatchSynthesisModule_MoveContainer, BatchSynthesisModule_AddSolution,
BatchSynthesisModule_Stir, BatchSynthesisModule_Heat, BatchSynthesisModule_Mix,BatchSynthesisModule_React,
answer may follow as below example script. Please refer below example code.
```

Examples for few-shot learning

````
**BatchSynthesisModule.py**
```python
from pydantic import BaseModel, Field
from typing import Dict, Any, Optional, Union

class BatchSynthesisModule_MoveContainer_FromTo(BaseModel):
    Type: str = ""

class BatchSynthesisModule_MoveContainer_Container(BaseModel):
    Type: str = ""

class BatchSynthesisModule_AddSolution_Material(BaseModel):
    Type: str = ""

class BatchSynthesisModule_AddSolution_Volume(BaseModel):
    Value: Union[int, float] = 0
    Dimension: str = "µL"

class BatchSynthesisModule_AddSolution_Concentration(BaseModel):
    Value: Union[int, float] = 0
    Dimension: str = "mM"

class BatchSynthesisModule_AddSolution_Injectionrate(BaseModel):
    Value: Union[int, float] = 0
    Dimension: str = "µL/s"

class BatchSynthesisModule_Heat_Temperature(BaseModel):
    Value: Union[int, float] = 0
    Dimension: str = "ºC"

class BatchSynthesisModule_Mix_Time(BaseModel):
    Value: Union[int, float] = 0
    Dimension: str = "sec"
class BatchSynthesisModule_Centrifugation_Power(BaseModel):
    Value: Union[int, float] = 0
    Dimension: str = "rpm"

class BatchSynthesisModule_Centrifugation_Time(BaseModel):
    Value: Union[int, float] = 0
    Dimension: str = "sec"
```
````

```
class BatchSynthesisModule_Centrifugation_Power(BaseModel):
    Value: Union[int, float] = 0
    Dimension: str = "rpm"

class BatchSynthesisModule_Centrifugation_Time(BaseModel):
    Value: Union[int, float] = 0
    Dimension: str = "sec"

class BatchSynthesisModule_Sonication_Power(BaseModel):
    PowValue: Union[int, float] = 0
    Dimension: str = "kHz"

class BatchSynthesisModule_Sonication_Time(BaseModel):
    Value: Union[int, float] = 0
    Dimension: str = "sec"

class BatchSynthesisModule_MoveContainer_Data(BaseModel):
    FromTo: BatchSynthesisModule_MoveContainer_FromTo
    Container: BatchSynthesisModule_MoveContainer_Container
    Device: Dict[str, Any]

class BatchSynthesisModule_AddSolution_Data(BaseModel):
    Material: BatchSynthesisModule_AddSolution_Material
    Volume: BatchSynthesisModule_AddSolution_Volume
    Concentration: BatchSynthesisModule_AddSolution_Concentration
    Injectionrate: BatchSynthesisModule_AddSolution_Injectionrate
    Device: Dict[str, Any]

class BatchSynthesisModule_Heat_Data(BaseModel):
    Temperature: BatchSynthesisModule_Heat_Temperature
    Device: Dict[str, Any]

class BatchSynthesisModule_Mix_Data(BaseModel):
    Time: BatchSynthesisModule_Mix_Time
    Device: Dict[str, Any]

class BatchSynthesisModule_Centrifugation_Data(BaseModel):
    Power: BatchSynthesisModule_Centrifugation_Power
    Time: BatchSynthesisModule_Centrifugation_Time
    Device: Dict[str, Any]
```

Figure R2. Prompt engineering of Pydantic class generation for type validation using the OpenAI API. The figure illustrates three steps of prompt engineering: role assignment, prompt definition, and examples for few-shot learning in generating Pydantic models for chemical tasks within a MAP setting.

After generating Pydantic classes for each task, we implement data type validation logic for correct task execution in the action translator, as illustrated in Figure R3. If a client inputs the wrong data type, such as an integer instead of a float, the Pydantic class will raise a system error and shut down the master node. Therefore, this type validation logic prevents human errors from invalid data types, ensuring safe task execution.

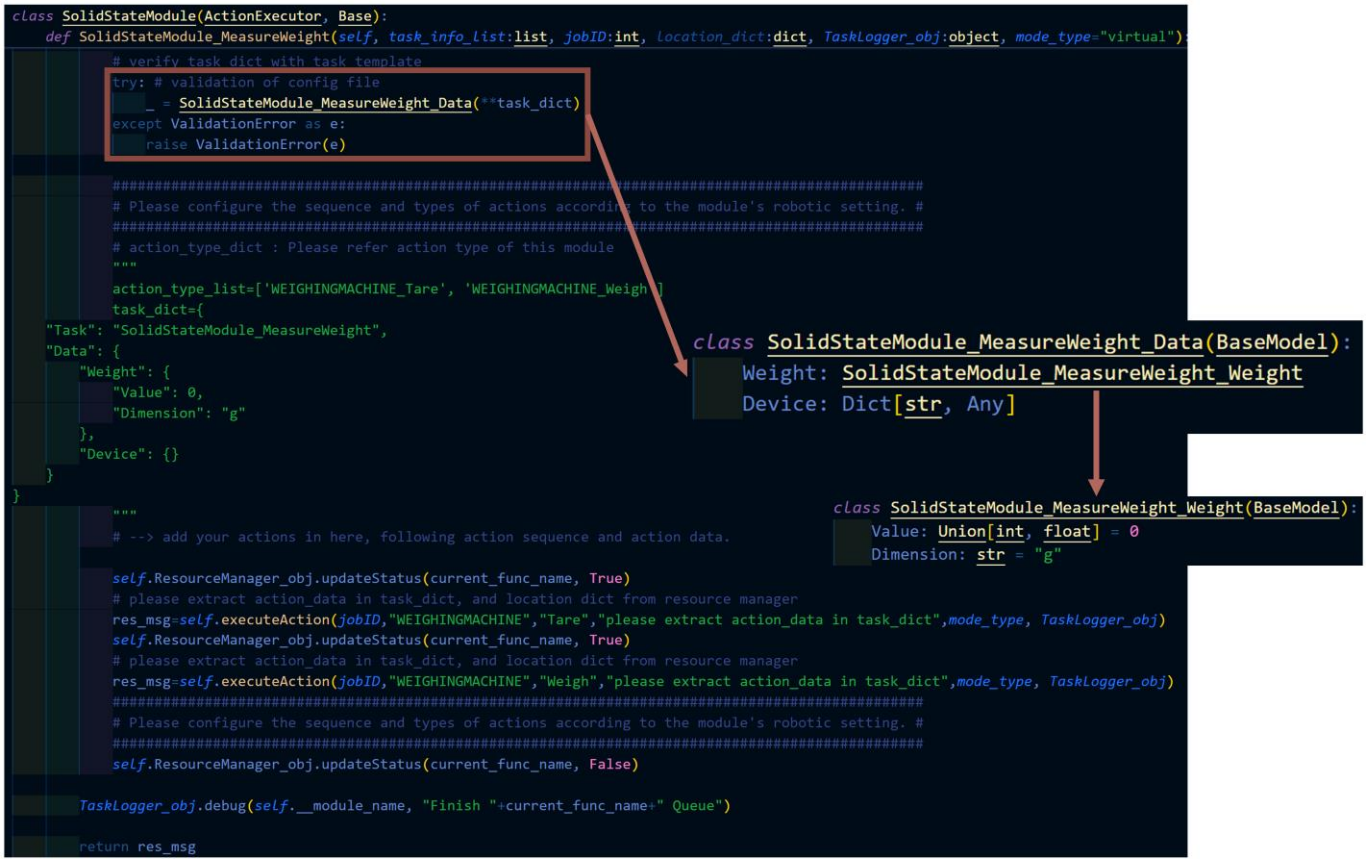


Figure R3. Code generation results of type validation logic in task functions of the action translator. During code generation via the Copilot of OCTOPUS, each task function in the action translator executes type validation logic before action sequence execution.

Additionally, we understand the reviewer's concern regarding the schema-less document database, and let me explain why we've chosen this format. Chemical experiments are influenced not only by experimental conditions but also by the experimental sequence[2]. Therefore, MongoDB, a non-relational database, can store the experimental order using a list data structure. On the other hand, relational databases cannot reflect such experimental sequence. While querying and utilizing the accumulated material data from a non-relational database may be slower, further research is needed to explore methods for converting this data into a relational database format.

[Revision to the manuscript] We modified some parts in the manuscript and **Supplementary Information**, as follows.

(Results – p15) "...In this stage, the data type validations are generated to check for invalid data in the task template, using "Pydantic" to predefine the JSON data type (Fig. 9d and Supplementary Figure S34)...

(Discussion – p16) "...Additionally, an advanced relational database management system for MAP should be developed to accommodate differently-formatted material data using JSON, as specified by various chosen modules...

(Supplementary Information) Figure S34. Example of task template and type validation generation

[References]

[1] https://github.com/pydantic/pydantic

[2] Ojea-Jiménez, Isaac, Neus G. Bastús, and Victor Puntes. "Influence of the sequence of the reagents addition in the citrate-mediated synthesis of gold nanoparticles." *The Journal of Physical Chemistry C* 115.32 (2011): 15752-15757. https://pubs.acs.org/doi/full/10.1021/jp2017242

[Comment 4] "3) Consider using more robust existing subsystems.

For example, the user management system is a single-file database that stores usernames and passwords in plain text. It would be difficult to extend this system to one with permissions and API keys that allow different users to perform different actions.

There are many solutions, including Auth0 as well as any major cloud provider, that have robust user management systems with modern authentication, authorization, and role-based access built in. Development time is generally better spent incorporating such pre-existing systems than recreating similar functionality within the project."

[Our response] We thank the reviewer for this comment. We recognize the limitations of the current login process, which is based on the SHA-256 encryption method and the sqlite3 database. This outdated login process would be challenging to extend in OCTOPUS to include permissions and API keys that allow different clients to perform diverse experiments.

Therefore, we implemented a modern authentication system using the Auth0 API. As shown in Figure R4, the login process with Auth0 begins with the user inputting their ID and password into the interface node. This node then requests authentication from Auth0, which handles the authorization and initiates a session. Once authenticated, the session is established, and the client gains access to the system, including the master node and module nodes.

Connecting to OCTOPUS involves the user inputting their ID, password, and job script into the interface node. The interface node then establishes a session and communicates with Auth0 for authentication. Upon successful authentication, the session allows access to the master node, which coordinates with the module nodes to process the client's job. This new login process with Auth0 ensures that OCTOPUS can be extended for large-scale MAP, allowing different clients to perform diverse experiments.

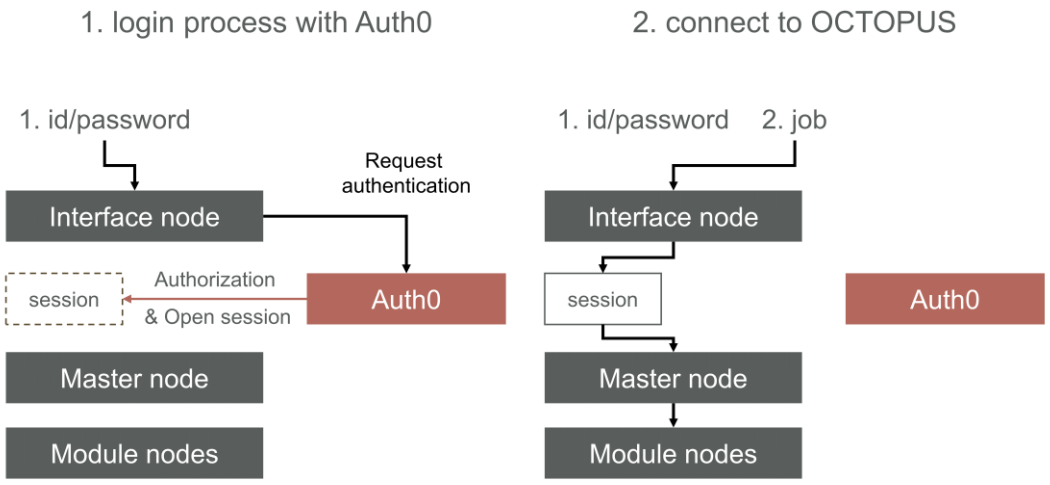
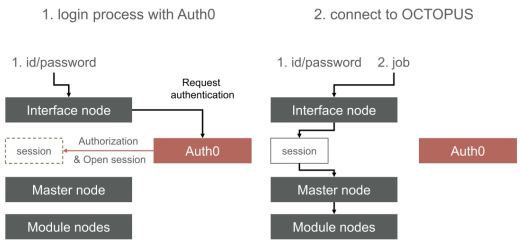


Figure R4. Workflow of the login process with Auth0. The Auth0 login process starts with the client entering their ID and password into the interface node. After successful authentication, a session is initiated, giving the client access to the master node and module nodes. The client can then submit a job script via the interface node. Once authenticated, the session permits access to the master node, which orchestrates the module nodes to execute the client's job.

[Revision to the manuscript] Following the reviewer's suggestion, we accordingly modified some parts in Results, as follows. We also added Figure S3 in **Supplementary Information**.

(Results, Page 6) "*The interface node is responsible for managing the client login process, enforcing stringent security policies through Auth0 API (application programming interface) functions while awaiting client commands (Supplementary Figure S3). Once authenticated, …*"

(Supplementary Information – Figure S3)



Responses to Reviewer Comments

Reviewer 4

[**General review**] "This work is a tour de force in establishing a code base coupled with detailed demonstrations of its deployment for automated materials science experiments. The authors describe various complementary works from the community, although I don't believe any published work covers the breadth of the present submission. I recommend this work for publication with very minor revisions, which are noted below after my assessment of how the reviewers addressed the comments from the previous review:

Reviewer 1 has noted the lack of "scientific findings", although I see this paper as a clear outline of how most scientific findings will be generated in labs that fully leverage the power of automation and AI. I think this work is quite visionary and I can speak firsthand on the technical difficulty of developing such a system. I think the combination of technical advance and vision make the work suitable for publication in Nat Comm.

Reviewer 2 noted the potential to improve the manuscript via discussion how "extended reality" may be leveraged in OCTOPUS. The authors nicely implemented this suggestion.

The comments of Reviewer 3 have been addressed via cleanup of the git repository, addition of the copilot, implementation of pydantic models, and implementation of Auth0.

I want to especially note that the addition of the co-pilot elevates the work compared to the initial submission. There has been lots of buzz around the use of LLMs for experiment design, and this work shows how the OCTOPUS infrastructure can capitalize on this opportunity.

[Our response] We thank the reviewer for the positive evaluation of our work.

[Comment 1] "In the abstract when the copilot is introduced with the phrase "ensure the reusability of OCTOPUS", I think "ensure" is an overstatement and can be replaced with, for example, "promote"..."

[Our response] We appreciate the constructive feedback on overstatement term (ensure).

[Revision to the manuscript] Following the reviewer's suggestion, we modified abstract parts to reflect "promote" term.

*(Abstract) "... Copilot of OCTOPUS is developed to **promote** the reusability of OCTOPUS for potential users with their own sets of lab resources, which substantially simplifies the process of code generation and customization through GPT recommendations and client feedback. ..."*

*(Discussion-p18) "... Copilot of OCTOPUS is provided to **promote** the reusability of OCTOPUS for potential users, which significantly simplifies the process of code generation and customization with GPT recommendations and client feedbacks. ..."*

[Comment 2] "For the "Network protocol-aided process modularization" in Fig. 4. To my knowledge, the first demonstration of this type of functionality was the following paper, which the authors may consider discussing and citing:

Vogler, M.; Busk, J.; Hajiyani, H.; Jørgensen, P. B.; Safaei, N.; Ramírez, F. F.; Carlsson, J.; Pizzi, G.; Clark, S.; Bhowmik, A.; Stein, H. S. Brokering between Tenants for an International Materials Acceleration Platform."

[Our response] We thank the reviewer for the valuable suggestion. As per the recommendation, we have discussed and cited the work by *Vogler et al.* in the revised manuscript. The relevant reference has been added to provide context for the network protocol-aided process modularization in Fig. 4.

[Revision to the manuscript] Following the reviewer's suggestion, we cited *Vogler et al* paper in the Results section (sub-section of network protocol-based modularization) to discuss about network protocol-based technique in MAP.

(Discussion- p8) " The network protocol was implemented for modularization, which have also been demonstrated in the work of Stein and coworkers[36] for facilitating efficient communications in a brokering system named FINALES (fast intention-agnostic learning server)."